Co-evolution model for data sources and views [*]

© Alexander Dolnik Sergei Syntylskiy

Saint-Petersburg State University
The Faculty of Mathematics and Mechanics
{alexander.dolnik, ssyntulsky}@gmail.com

Abstract

ETL process evolution is investigated below. A model-driven approach to templates and ETL process evolution problem is developed. We suppose that the ETL process evolution problem is mainly a problem of a low abstraction level. So the definition of ETL process based on a conceptual model is a principal step towards effective ETL evolution. Our approach seems to be scalable, robust and simpler in use compared to existing ETL evolution frameworks and tools.

1 Introduction

From day to day new applications of data warehouses are discovered. A data warehouse is integrated, time-varying, subject-oriented, non-volatile collections of data that is used primarily in organizational decision making.

But sometimes world is changed by various causes. Some of these causes are consequence of following factors: juristical, industrial and technological, financial, analytical and many others.

So we need to support ability for data warehouse model evolution.

Extract-transformation-load (ETL) process connects data warehouse with external data sources. ETL process and data warehouse are to change when data source schema has changed. Sometimes this changes may produce difficulties for developers and administrators. There are a lot of frameworks (for example, [11], rails, ARKTOS II, Scriptella), but framework solutions may be hard to support and costly to implement. Usually developer divides an ETL process in several stages. A stage can be implemented using special tools and methods. Such decoupling makes the whole process more robust.

We concentrate on an user-friendly and clear design of the ETL process for considered scope of a problem.

The rest of this paper is organized as follows. In Section 2 problem definition and motivation example are presented. The related works are observed in Section 3. A co-evolution model for data sources and views of data warehouse is proposed in Section 4. Following in Section 5 we present this model as applied to our example. We conclude with directions for future work in Section 6.

* This work was partially supported by RFBR (grant 07-07-00268a).

Proceedings of the Spring Young Researcher's Colloquium on Database and Information Systems, Saint-Petersburg, Russia, 2008

2 Problem definition

Let us consider an analytical information system used in a number of state regulating institutions. It is intended for automation of document flow between a regulating institution and subjects under regulation and analysis of collected data.

A natural monopoly regulation is an important economical function of the state. The regulation involves an intensive document flow between a regulating institution and organizations under control. Today it is a highly-automated process based on a set of formalized templates. The templates are filled by actual data of organizations, for example, expenses or production volume. Then the data from templates are loaded into the regulating institution data warehouse.

The basic source of data for the system are templates filled by organizations under regulation. Templates are forms for data input implemented as MS Excel tables. The template is affirmed by the regulating institution and then published. The template has following characteristics:

- list of organization properties included into the template, organizations under regulation must give values of this properties;
- periodicity of data collection using this template: if the template must be filled repeatedly, organizations are obliged to give new data on a regular basis, for example, yearly or quarterly.

See an example of a template on Figure 1.

When the template is published, officers of the adjustable organizations are responsible for filling the template. The filled templates are sent back to the regulating institution. They are validated automatically and, if necessary, manually, then or are accepted, else are sent back for filling again. For each template a table in the regulating institution database is generated automatically. Such tables contain a column for each property of the corresponding template. Data of templates are loaded into these tables: properties values, template metadata containing name of organization, region and time period. After templates data are loaded into a database they are used by various applications for viewing and analyzing

Property	Value
1. Lubricants	
2. Salary	
3. Repair	
4. Depreciation	
5. Nonproduction expenses:	
5.1 Credit expenses	
5.2 Training expenses	
...	
10. Necessary Gross Receipt	
...	
15. Product 1	
16. Product 2	
...	

Figure 1: Template v1

of the data, building reports and analytical cubes. The templates change rather frequently. This changes concern both structure of collected parameters (usually collected data expands) and names of the parameters already included in a template. For example, the template shown on Figure 1, can evolve to one shown on Figure 2 – in the new version expenses on lubricants are detailed. The expenses on the salary are separated into salary of employees who are participating and not participating in manufacture. The production of kind 2 is divided into two subspecies – 2_1 and 2_2.

Besides the structure of a template the period of data collection can change. Changes of template are treated as a combination of deactivation of old template and adding new ones. So the old table in which data were collected using previous version of template does not change, and a table for new template is added. It is obvious, that though the structure of template evolves, applications and views based on previous versions of templates should work in the same way with new versions of templates where it is possible. Thus, the problem is offering a method of automatic (or semi-automatic) co-evolution of views and templates. Some views constructed on the basis of example templates are listed below:

1. The view "Organization expenses" contains detailed information on structure of expenses of the organizations in the accounting periods. For this view inclusion of the greatest possible set of parameters is desirable. Thus the situation in which value of some parameters for some periods is not known must be considered. This view should contain all parameters concerning the expenses even included only in one version of the template.

2. The dynamic of production 2 is in the whole state. This view should display value of annual production 2 of the first version of the template and the production values sum of kinds 2_1 and 2_2 summed for four quarters from second version.

3. The effectiveness of product 1 in various regions. Calculation of a product effectiveness of volumes and expenses is a complex procedure. There are various ways to divide constant expenses, such as rental costs of administrative office, on various kinds of production. In this article we treat this procedure as a black box. This view, probably, must not change when template evolves.

Property	Value
1. Lubricants	
1.1 Fuel	
1.2 Diesel	
1.3 Other	
2. Salary of production employees	
3. Social expenses	
4. Repair expenses	
5. Depreciation	
6. Nonproduction expenses:	
6.1 Credit exps	
6.2 Training exps	
6.3 Salary of nonproduction exps	
...	
10. Necessary Gross Receipt	
...	
15. Product 1	
16. Product 2_1	
17. Product 2_2	
...	

Figure 2: Template v2

The example of transition from one version of template to another has been considered. Three views using templates' property values should be differently processed:

- new columns should be added to representation "Organization expenses",
- calculation of production dynamics will become a little more complicated for data of second version template due to reduction of data collection period and splitting of production 2 on subtypes,
- the view, containing the information on the effectiveness, does not depend on the evolution of patterns.

In a Figure 3 the schema of a system's part under consideration is given. In the figure connections between properties, views and templates are shown as arrows.

Aforementioned example illustrates typical problems that ETL process developer faces with. Today there is no general solution for this problem. Practice shows that there is not "silver bullet". Developers usually avoid using these frameworks because of their complexity. Often evolution problems can be solved by using correct and well considered conceptual design of schemas and ETL processes.

The main aim of this paper is to develop evolution model for aforementioned case and demonstrate how it can help to construct flexible and stable ETL process.

3 Related work

Some methods are developed for ETL processes and data warehouse design and modelling (for example [5]). In

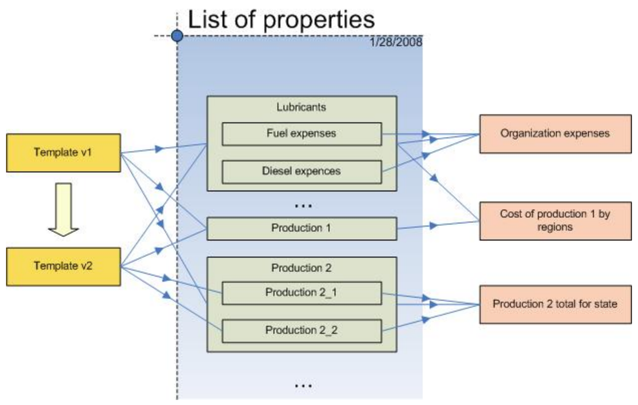


Figure 3: Scheme of Conceptual Level. Motivation Example

work [5] framework for data description and ETL process design by using extend of traditional UML is suggested. Moreover, OMG defines MOF-based standard for modelling warehouses (CWM, [2]). Instead of OMG modelling methods EER or ontology-based languages can be used. Approach suggested in [8] divides conceptual and logical levels: the conceptual level (ontology defined in OWL *SemanticWeb*) is used for describing extern data sources and data warehouses, the logical level is used for describing ETL process. Logical level is defined with declarative language LDL++ ([13]). This language was chosen because of its expressiveness (for example, supported external function call).

In work [10] semi-automatic framework is developed. The main idea is to define a number of template blocks and build a process of them.

The main problem of aforementioned frameworks is absence of universal approach to connecting conceptual level with logical. Another evolution method is suggested in [6]. This method is based on ETL process graph construction. Each graph node corresponds to transformation model element and is annotated with What-If policies, containing instructions for changing graph structure.

However, this approach is difficult to implement, because of enormous amount of additional information required in nontrivial cases.

In article [1] the authors propose technical solution of problem in question applied to analytical cubes. However it does not cover the whole ETL process evolution.

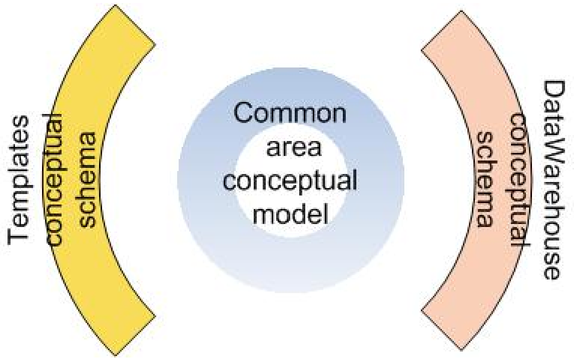
It is possible to describe evolution methods based on axioms [7, 9]. Actually axiomatic approach can be used to formalize a part of the problem under consideration, but fails to solve it as a whole.

4 Common conceptual model

In this section we suggest model of views and templates co-evolution. First of all we divide each ETL process into three levels:

- templates conceptual schema represents a set of input data sources schemas;

- data warehouse conceptual schema represents a data warehouse schema that necessary for data mining or for support decision making applications;



Templates conceptual schema
Common area conceptual model
DataWarehouse conceptual schema

Figure 4: Scheme of Common Conceptual Model

- common area conceptual model represents an ETL process model (mapping templates into data warehouse).

This construction depicted on Figure 4.

Then we describe common area conceptual model metadata in natural way using the EER model suggested by Bernhard Thalheim in [3, 4, 12]. Schema of the whole system is depicted on figure 3.

Now we have got common area conceptual model in EER model terms. A lot of drag-and-drop tools exist for easy (visual) mapping concepts in the common area conceptual model to concepts in templates conceptual schema. The second task is to describe mathematical conditions and equations for the common area conceptual model. Mathematical conditions and equations include following elements:

- Aggregation functions. There are three types of aggregation functions: distributive, algebraic, holistic. Distributive and algebraic aggregation functions are investigated in Thalheim's work in [3, 4]. A structural recursion is used for their description. Holistic functions are under investigation. But this class of aggegation funsctions is not mentioned in article.

- Group operations. They will be described in section 4.1).

- Time constraints. They will be described in section 4.2.

- Transformations. Properties of transformation functions can be found in work [4]. Details explanation is not included into this article.

The third question is to map concepts in the common area conceptual model to the data warehouse conceptual schema. We developed high-level view declaration language for this purpose (will be described in section 4.1). This language must be stable to changes in templates set. Also it must be flexible for describing transformation with wide diapason of different templates. We suppose engine for looking through templates set and generating SQL code for templates based on our view declaration.

4.1 View declaration

Our model has the following basic and extended modelling constructs:

1. Set of entities, actually in our case it is enough to consider only one entity – Organization.
2. Set of simple properties, each property has name and domain.
3. Groups, group is a complex property that contains another groups or simple properties. All properties contained in a group (immediately or transitively) must have the same type.
4. Constraints. Constraints can be defined for each group or property. A corresponding logical operator can be defined for each type. A set of logical formulas using this operator can define the integrity constraints which are valid for each instance of the type.
5. Operations, defined for each type.

Typical (in relational databases) views are created by execution some query like:

create view name (projection variables) **as**
  select projection expression
    from database sub-schema
    where selection condition
    group by expression for grouping
      having selection among groups
    order by order within the view;

But this is not convenient for our purposes. We need to develop new view based on our model. Following extensions of EER model must be taken into account: data temporality, hierarchical types, schema modularity, calculated values. These extensions will be described below in details. Generally we have auxiliary schema $\mathcal{A}$. See [12] for details of auxiliary schema construction method. New view is defined on top of an EER schema by

- a schema $\mathcal{V} = \{S_1; \ldots; S_m\}$, where S_i is a target type,
- an auxiliary schema $\mathcal{A}$ mentioned above and
- a query $q : \mathcal{D} \times \mathcal{A} \rightarrow \mathcal{V}$, where $\mathcal{D}$ is a given database.

Generalized view schema suggested by Thalheim is depicted below:

generate Mapping : Vars $\longrightarrow$ output structure
  from database types
  where selection condition
  represent using general presentation style
    & Abstraction (Modularity, measure, precision)
    & Orders within the presentation & Points of view
    & Hierarchical representations & Separation
    & Temporality
  browsing definition condition & Navigation
  functions Search functions & Export functions
    & Input functions & Session functions
    & Marking functions

We simplified this generalized view schema for our needs. And special language for construction views has been developed. It has the following Backus-Naur notation (BNF):

generate Mapping $::=$ *properties* **properties**,
  $[hierarchical\ type]$, [**order by**], [**where**]
hierarchical type $::= ((level[modifiers_2])^*, scope)^+$
time $::= (period[modifiers_2])^* timeset$
properties $::=$
$(property[modifiers_1][\{$**properties**$\}])^*$
modifiers$_1$ $::= ([show][sum_{ignore}, sum_{undefined}])$
modifiers$_2$ $::= mandatory$

Time can be represented as a hierarchical type. A **generate Mapping** is parsed to SQL code by finding appropriate templates, aggregation, sorting and filtering. Detail parser description is omitted in this work.

At the section 5 our example will be described in detail using aforementioned BNF.

The next section we will discuss temporal aspects of our model.

4.2 Temporal properties for templates and views

Suggested procedure of views change or creation contains the following steps:

1. The User selects parameters of the Organization from the list of available parameters, including both directly collected, and calculated;
2. Using given list of parameters and the list of possible periods of data collection (year, quarter or month) program constructs pairs of time sets and periods such as all necessary parameters are collected with this period within given time set;
3. The User selects the period and the time interval, being a subset of the set constructed for the period on the previous step;
4. Program automatically creates or updates view, including data on parameters chosen on step 1 with the period and the restrictions set by the user on step 3.

In the following part of the current section we introduce method allowing to construct pairs of periods and temporal constraints using temporal annotations of templates. These annotations are automatically created as a result of start and stop of data collection. This method implements step 2 of the procedure. For each template it is possible to define a set of pairs, containing period of data and time when data were being collected with this period. The period of data collection is a year, a quarter or a month. The time when data were being collected with given period can be represented as union of several intervals. Thus borders of intervals should represent points, multiple to the period of data collection. For example, if data are collected yearly, the interval should begin and end on the first of January.

It must be noticed, that as the considered periods of data collection form a hierarchy. It is possible to treat time intervals as disjunctive. For example, some time the template data are being collected simultaneously with the periods equal to month and to quarter. If suddenly it appears quarterly collections can be ignored. However such situation is hardly probable in practice.

We use following notation: $Hold(t, p)$ means time set when data of template t is collected with period p. So, because periods are hierarchically ordered, if interval p is shorter than q then statement $Hold(t,p) \subset Hold(t,q)$ holds. $Vars(x)$ means set of properties, used by x, where x is a template or view. So the time set of view v for period p can be calculated using the following formula:

$$Hold(v,p) = \wedge_{u \in Vars(v)} \vee_{\{t | u \in Vars(t)\}} Hold(t,p).$$

Thus, a method of calculating temporal characteristics of view on the basis of information about templates data collection is introduced.

4.3 Constraints and hierarchical types

As it was mentioned above hierarchical types and schema modularity (groups) must be constructed. Firstly formal model need to be determined on subject domain. Then using this formal model we will be able to check correctness of our construction.

Thereto, following hierarchical data types are introduced.

It can be defined on base types, but with the following extensions. Base type $B = (Dom(B), Op(B), Pred(B), \Upsilon)$ is extended with predicate set $Pred(B)$ and constraint set Υ. Predicates $Pred(B)$ define a number of equivalence relations eq on domain $Dom(B)$. Each of these equivalence relations define a partition Π_{eq} of the domain into equivalence classes. For each equivalence class c of partition Π_{eq} we introduce a name n_c. This partition with named classes can be denoted by Π^*.

There are two trivial named partitions that only relate elements to themselves is denoted by $\perp^*$ and that consists of $\{Dom(B)\}$ is denoted $\top^*$.

Equivalence relations and partitions may be ordered. The canonical order of partitions on $Dom(B)$ relates two partitions Π^*, Π'^*. We define $\Pi^* \preceq \Pi'^*$ if and only if for all (c, n_c) from Π^* there exists one and only one element $(c', n_{c'})$ from Π'^* such that $c \subseteq c'$.

If it is necessary, we can also consider non-classical orderings such as the majority order $\preceq_m^{choice}$ that relates two named partitions. In our example canonical order is enough.

For instance, we can define types hierarchy for time and volume types.

According group definition we can declare type hierarchy for whole group.

The next step is to add elementary evolution transformation into group operations:

- adding new property/group;
- deleting useless property/group;
- transferring property up/down over group hierarchy.

Before executing operation model constraints are to be checked. Constraints may be manually added or automatically obtained from other constraints.

For example, we may define constraints for introduced groups operations. According to the work [3] when aggregation functions are defined for group the group operations became restricted in natural way.

Consequently using mentioned above techniques(group definition, aggregation function declaration, hierarchical types and temporal properties definitions) it is possible to achieve view schema construction and data evolutions with minimal efforts from developers and database administrators.

Now we can describe extended view as it was proposed in Section 4.1 with our auxiliary schema $\mathcal{A}$.

5 Real example

We defined two types of summarize functions: $sum_{undefined}$, sum_{ignore}. The $sum_{undefined}$ function will be "undefined" if at least one parameter is equal to "null" or "undefined". The sum_{ignore} function will summarize values, but "null" and "undefined" values are ignored. In view declaration we explain how to use groups by binding them with summarize functions. We need to declare data hierarchy:

$$date = \{partition_{quarter}, partition_{year}\},$$

$$partition_{quarter} \subseteq partition_{year}$$

For example, there are four organizations: ORG1, ORG2, ORG3, ORG4. Each organization uses its own template. The first organization ORG1 collects data of SocExp and RepExp expenses every quarter. The second organization ORG2 collects data of SocExp and RepExp expenses every year. The third organization ORG3 collects data of RepExp expenses only but every quarter. The fourth organization ORG4 does not collect any expenses group data.

Senior analyst want to obtain detail summary of organizations expenses in 2001-2002 years.

Using our framework he has to write the query like:

generate Mapping "View Organization expenses" :
Expenses_group($show\ sum_{ignore}\ as$ **SumE**),
Org, $date\big(($**Quarter**, **Year** $mandatory), [2001 : 2002]\big)$
Order by *Org, date*

The following table will be produced:

Org	Year	Quarter	SocExp	RepExp	SumE
ORG1	2001	1	23	3	26
ORG1	2001	2	24	3	27
ORG1	2001	3	NULL	7	7
ORG1	2001	4	23	10	33
ORG1	2001	ALL	70	23	93
ORG2	2001	ALL	80	13	93
ORG3	2001	1	UnDef	5	5
ORG3	2001	2	UnDef	7	7
ORG3	2001	3	UnDef	6	6
ORG3	2001	4	UnDef	6	6
ORG3	2001	ALL	UnDef	24	24

Upon supposition that before 2002 year organization ORG1 used to gather data every year. In 2002 year organization ORG1 started to collect data every quarter. Also in 2003 year organization ORG1 splitted quantity of Prod2 into two parameters: Prod2.1 and Prod2.2.

Group **Product2_group** evolved in common area conceptual model. At first it had contained only Prod2 parameter. Then Prod2 was splitted into Prod2.1 and Prod2.2. And equation "$sum_{ignore}(Prod2.1, Prod2.1) = Prod2$" was added into mathematical conditions of common area conceptual model.

The following query helps to construct summary report for ORG1 in 2001-2003 years:

generate Mapping "View Quantity of product 2" :
Product2_group$(show\ sum_{ignore}\ as$ **Prod2**$)$,
Org, $date\big(($**Quarter**, **Year** $mandatory)$,
$[2001 : 2003\ years]\big)$
Order by *date, Org*

The following table will be produced:

Org	Year	Quarter	Prod2	Prod2.1	Prod2.2
ORG1	2001	ALL	80	UnDef	UnDef
ORG1	2002	1	10	UnDef	UnDef
ORG1	2002	2	8	UnDef	UnDef
ORG1	2002	3	15	UnDef	UnDef
ORG1	2002	4	10	UnDef	UnDef
ORG1	2002	ALL	43	UnDef	UnDef
ORG1	2003	1	13	3	10
ORG1	2003	2	14	7	7
ORG1	2003	3	14	8	6
ORG1	2003	4	9	7	2
ORG1	2003	ALL	50	25	25
...	...	...	...	...	...

The following example requires to define hierarchy of organization region and black box functions for calculation average effectiveness in a region. Value of the black box function **EffectCalcFunc** for a region equals average value of $\frac{product\ cost}{expenses}$ fractions for each organization in the region.

View constructions has became more complex in calculations but declaration remains clear and short.

generate Mapping "Effectiveness of organizations in regions" :
EffectCalcFunc$(Product\ Cost,$ **Expenses_group**$)$
$Org(region\ mandatory)$,
$date\big(($**Quarter**, **Year** $mandatory), [2001 : 2002]\big)$
Order by *region, date*

The following table will be produced:

region	Year	Quarter	EffectCalcFunc
SPb	2001	1	34,23
SPb	2001	2	32,53
SPb	2001	3	35,67
SPb	2001	4	30,00
SPb	2001	ALL	33,34
SPb	2002	ALL	31,89
LenObl	2001	1	23,05
LenObl	2001	2	18,17
LenObl	2001	3	24,67
LenObl	2001	4	23,45
LenObl	2001	ALL	22,13
LenObl	2002	1	25,34
LenObl	2002	2	24,10
LenObl	2002	3	25,07
LenObl	2002	4	26,08
LenObl	2002	ALL	25,63

6 Conclusion

We developed a conceptual model for considered data warehouse metadata and used this model to define ETL process. Definition of ETL process based on conceptual model is more abstract than sql-based one, proper abstraction level helped us to keep off many problems both with schema evolution and consistency maintainency. Suggested model is based on EER model developed by Bernhard Thalheim. This method allows to escape versioning and damping evolution. A method developed in this paper has been applied to data warehouse of natural monopoly regulating institution.

References

[1] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997.

[2] OMG group. *Common Warehouse Metamodel(CWM) Specification*, 2001), ee = .

[3] Hans-J. Lenz and Bernhard Thalheim. Olap databases and aggregation functions. In *SSDBM '01: Proceedings of the Thirteenth International Conference on Scientific and Statistical Database Management*, page 91, Washington, DC, USA, 2001. IEEE Computer Society.

[4] Hans-Joachim Lenz and Bernhard Thalheim. Olap schemata for correct applications. In *TEAA*, pages 99–113, 2005.

[5] Sergio Lujan-Mora, Panos Vassiliadis, and Juan Trujillo. Data mapping diagrams for data warehouse design with uml.

[6] George Papastefanatos, Panos Vassiliadis, Alkis Simitsis, and Yannis Vassiliou. What-if analysis for data warehouse evolution. In *DaWaK*, pages 23–33, 2007.

[7] Randel J. Peters and M. Tamer Ozsu. An axiomatic model of dynamic schema evolution in objectbase systems. *ACM Trans. Database Syst.*, 22(1):75–114, 1997.

[8] Timos K. Sellis and Alkis Simitsis. Etl workflows: From formal specification to optimization. In *ADBIS*, pages 1–11, 2007.

[9] A. Simanovsky. Evolution of schema of xml-documents stored in a relational database. In J Barzdins, editor, *Proc. of the Baltic DBIS'2004*, volume 672, pages 192–204, Riga, Latvia, June 2004. Scientific Papers University of Latvia.

[10] Alkis Simitsis. Mapping conceptual to logical models for etl processes. In *DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 67–76, New York, NY, USA, 2005. ACM.

[11] Darja Solodovnikova. Data warehouse evolution framework. In *Proc. of the SYRCoDIS'2007*, Moscow, Russia, June 2007.

[12] Bernhard Thalheim. *Entity-Relationship Modeling, Foundations of Database Technology*. Splinger, 2000. XII, 627 pp. 160 figs., Hardcover.

[13] Carlo Zaniolo. *LDL ++ Tutorial*, 1998), ee = http://www.cs.ucla.edu/ldl/tutorial/ldlcourse.html,.